How to Automatically Understand and Integrate System-models ... and how SpecIF can help.

Oskar von Dungern, Dr.-Ing., adesso AG
Topics Today

1. The idea behind model integration ... and SpecIF
   - Purpose
   - Use-Cases
   - Principles and Approach

2. An example

3. Positioning ReqIF, OSLC and SpecIF

4. Status
Purpose: SpecIF integrates information from different sources
Diverse systems name information elements differently

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object Short Text:</strong> Crisp Naming</td>
<td><strong>Title:</strong> Crisp Naming</td>
</tr>
<tr>
<td><strong>Object Text:</strong> Longer description as formatted text, sometimes with a diagram.</td>
<td><strong>Description:</strong> Longer description as formatted text, sometimes with a diagram.</td>
</tr>
<tr>
<td><strong>Object Priority:</strong> 1_high</td>
<td><strong>Severity:</strong> 1_high</td>
</tr>
<tr>
<td><strong>Last modified on:</strong> 2017-09-12</td>
<td><strong>Modified Date:</strong> 2017-09-12</td>
</tr>
</tbody>
</table>
A vocabulary or ontology assigns a meaning to information elements and allows information exchange without negotiation.

The prostep ivip vocabulary for ReqIF defines some attribute names.
Use-Case: Exchange requirement specs between OEM and Supplier

OEM

- Requirement-Management

Supplier

- PLM/ALM

Standard ReqIF with
- prostep ivip vocabulary
Use-Case: Requirement Agreement using the HIS-Process

**Requirement**

<table>
<thead>
<tr>
<th>ReqIF.Name:</th>
<th>Crisp Naming</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReqIF.Description:</td>
<td>Longer description as formatted text, sometimes with a diagram.</td>
</tr>
<tr>
<td>Priority:</td>
<td>1_high</td>
</tr>
</tbody>
</table>

**Supplier-Status:**

**Supplier-Comment:**

**OEM-Status:** to-evaluate

**OEM-Comment:**

- An attribute pair for each of the suppliers
- An attribute pair for the OEM
Use-Case: Requirement Agreement using the HIS-Process

### Requirement

**ReqIF.Name:** Crisp Naming

**ReqIF.Description:** Longer description as formatted text, sometimes with a diagram.

**Priority:** 1_high

**Supplier-Status:**

**Supplier-Comment:**

**OEM-Status:** to-evaluate

**OEM-Comment:**

---

### Requirement

**ReqIF.Name:** Crisp Naming

**ReqIF.Description:** Longer description as formatted text, sometimes with a diagram.

**Priority:** 1_high

**Supplier-Status:** partly-agreed

**Supplier-Comment:** Comments or questions of the supplier

**OEM-Status:** to-evaluate

**OEM-Comment:**
Use-Case: Requirement Agreement using the HIS-Process

### Requirement

<table>
<thead>
<tr>
<th>ReqIF.Name:</th>
<th>Crisp Naming</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReqIF.Description:</td>
<td>Longer description as formatted text, sometimes with a diagram.</td>
</tr>
<tr>
<td>Priority:</td>
<td>1_high</td>
</tr>
<tr>
<td>Supplier-Status:</td>
<td>partly-agreed</td>
</tr>
<tr>
<td>Supplier-Comment:</td>
<td>Comments or questions of the supplier</td>
</tr>
<tr>
<td>OEM-Status:</td>
<td>not-accepted</td>
</tr>
<tr>
<td>OEM-Comment:</td>
<td>Reactions and Answers of the OEM</td>
</tr>
</tbody>
</table>

### Requirement

<table>
<thead>
<tr>
<th>ReqIF.Name:</th>
<th>Crisp Naming</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReqIF.Description:</td>
<td>Longer description as formatted text, sometimes with a diagram.</td>
</tr>
<tr>
<td>Priority:</td>
<td>1_high</td>
</tr>
<tr>
<td>Supplier-Status:</td>
<td>partly-agreed</td>
</tr>
<tr>
<td>Supplier-Comment:</td>
<td>Comments or questions of the supplier</td>
</tr>
<tr>
<td>OEM-Status:</td>
<td>to-evaluate</td>
</tr>
<tr>
<td>OEM-Comment:</td>
<td></td>
</tr>
</tbody>
</table>
Use-Case: Exchange model-based specs between OEM and Supplier

OEM
- Requirement-Management
- System-Modelling

Supplier
- PLM/ALM
- Simulation

Standard ReqIF with
- SpecIF Vocabulary
- SpecIF Assertions

Model Integration with SpecIF
Use Case: Exchange model information between tools

1. Change a state diagram in SysML
2. Check consistency of the specification
3. Validate the behavior in a simulation tool
4. Generate code for an embedded controller
What is SpecIF?

• „Specification Integration Facility“

• SpecIF adds conventions to express meaning to known technical formats such as ReqIF or OSLC.

1. Vocabulary for Objects, Relations and Attributes
   „Requirement“, „Actor“, „State“, „Event“ ...
   „satisfies“, „reads“, „contains“, „triggers“, ...

2. Logic Assertions
   („First-order predicate logic“)
   „An Actor satisfies a Requirement“
   „An Event triggers an Actor“

→ SpecIF carries both the „Visible“ and the „Meaning“
SpecIF carries both the Visible and the Meaning

Medium/Format
(The Presentation)

original, loss-less

Notation
(The Visible)

Interrelates all elements of all model diagrams and all notations

abstracted

Integrated Model
(The Meaning)

Objects and Relations
(The Persistence)
Abstraction: All model diagram types (notations) consist of three fundamental model-element types*

<table>
<thead>
<tr>
<th>Model-element Type</th>
<th>□ Model-diagram Type (Notation)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BPMN Business-process</td>
</tr>
<tr>
<td>Actor</td>
<td>X</td>
</tr>
<tr>
<td>State</td>
<td>(X)</td>
</tr>
<tr>
<td>Event</td>
<td>X</td>
</tr>
</tbody>
</table>

* Prof. Dr. Siegfried Wendt, Founding Director of the Hasso-Plattner-Institute, Potsdam: Fundamental Modelling Concepts
A complete specification needs also 'feature' and 'requirement'

<table>
<thead>
<tr>
<th>Model-element Type</th>
<th>BPMN Business-process</th>
<th>State-machine</th>
<th>System-composition</th>
<th>Organisation Chart</th>
<th>UML Classes</th>
<th>Document Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>X</td>
<td>(X)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>State</td>
<td>(X)</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Event</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Requirement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Taking a model diagram, automatically identify model elements and their relations to build a semantic net

Process Z [business process]

- Start Event!
- Process Step 1
- Process Step 2
- End Event!

- Process Z
  - User
    - Start Event!
  - Process Step 1
  - Process Step 2
  - End Event!

5 model elements identified

5+4+1+1+1 relations identified

triggers, follows, shows, contains, signals
Consolidate model elements from different diagrams

- Key to success is the abstraction using 5 fundamental model element types
- Impossible with 162 model element types in SysML and almost as many in BPMN
- Even within SysML the current tools fail to properly consolidate model elements from different model views
Summary: 4 steps to create an Integrated System Model

1. Identify
   Every element on a diagram
   Every line in a spread-sheet

2. Assign fundamental element type
   ■ Actor
   ● State
   ♦ Event
   ✶ Feature
   ✈ Requirement

3. Consolidate

4. Semantically Interrelate
   Component-070 contains Component-073
   Component-070 satisfies Requirement-4711
Add partial models step-by-step ... and use/deliver in known technical formats
Example: Search, navigate and audit in a common context
Positioning ReqIF, OSLC and SpecIF

- **Types + Instances**
  - ReqIF
  - OSLC
  - SpecIF

- **Format → Syntax**
  - Prostep ivip ReqIF Implementation Guide

- **Vocabulary for**
  - 1 Object Type (Requirement)
  - 0 Relation Types

- **Logic Assertions**
  - Semantics

- **Vocabulary for**
  - n Object Types
  - m Relation Types
Status

Notation mapped to ...

SpecIF
Open SE Models

... mapped to Persistence

Specification Integration Facility

FMC
SysML
Interaction Room
BPMN

ReqIF
SpecIF
OSLC
linked-data
WIKI
Document

available
available
available
available
available

Master Thesis at FhG IPK (Berlin)
Master Thesis at Paluno (Univ. Duisburg/Essen)
Master Thesis at HTW (Berlin) (starting 2017-10)
SpecIF Goals and Benefits

- Exchange model-based spezifications between tools and organizations.
- Combine (formatted) texts and models from different tools.
- Search, navigate, and check in a common context.
- Manage the product lifecycle from the beginning („end-to-end“):
  - Overarching engineering-disciplines
  - Combining methods
  - Technology-neutral
  - Vendor-neutral
  - Schema-conforming
  - Standard-conforming
  - Open and cooperative
Literature


Interesting?

oskar.dungern@adesso.de
Details for further discussion
The eye sees the same – SpecIF adds interrelated model data

Creating the „Visible“
- Text editing and image „drawing“
- Brain and Discipline to build and keep it consistent

Partial Modelling
- Text editing and modelling per method
- Tool support within the methods

Model Integration
- Text editing and modelling per method
- Elements in all views are interrelated by a semantic net
A SpecIF data set contains both the types („model“) and the instances („data“ = „payload“)

SpecIF model with Hierarchy-, Object- and Relation-types

The types can be defined at runtime („dynamic model“)

Hierarchical ordering of Objects (e.g. for convenient reading)

SpecIF data (payload) with Objects and Relations
The discussed SpecIF Model with 5 Fundamental Model-element Types is proven for Model Integration.